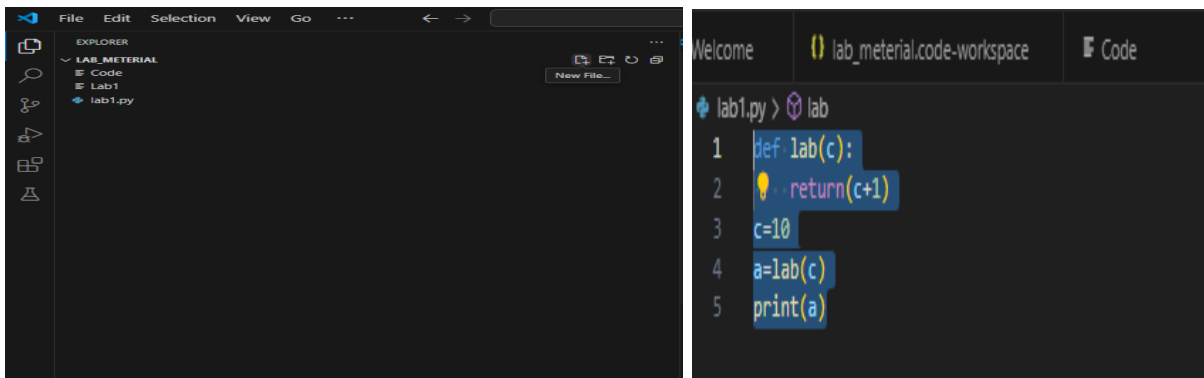


Lab 1 & 2 &3 &4

Fundamentals of Artificial Intelligence By Dr.Raghad Al-Shabandar

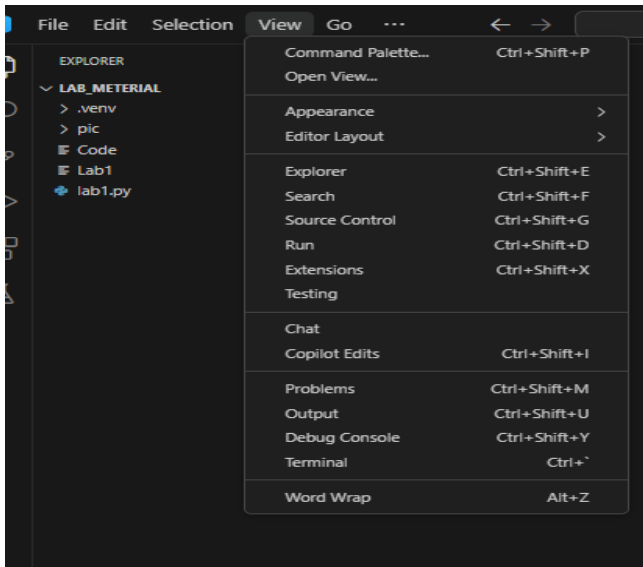
1.1 How to create python file

From the File Explorer toolbar, select the **New File** button on the lab_material folder:



1.2 Create a virtual environment

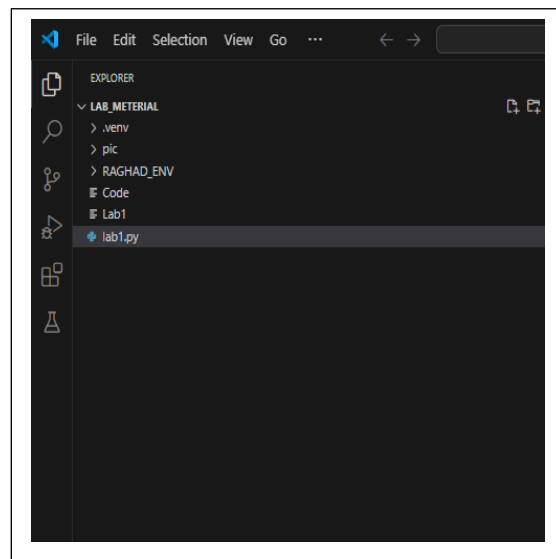
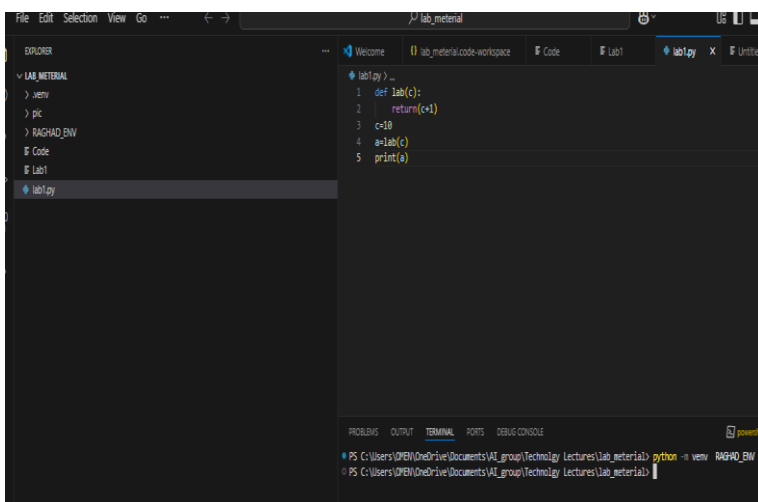
A virtual environment is a built-in tool for creating isolated Python environments. It generates a dedicated folder that contains a copy (or symlink) of a specific Python interpreter.



Steps

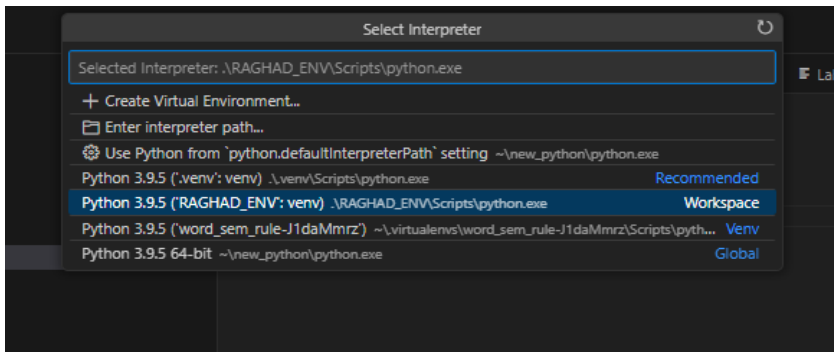
1: Open the Terminal from VS

- Write comments (`python -m venv envname`).
- The a new directory called `RAGHAD_ENV` in our project folder, containing the Python executable and necessary files for the virtual environment.
- See the contain of VR by writing the comments (`dir RAGHAD_ENV`)



2. To start using virtual environments, we should activate it. Go to the path of environment, and run command in the terminal:

- Change Directory to Scripts by writing comments (`cd Scripts`)
- Activate the Virtual Environment by writing comments(`activate.bat`)
- After activation go to command paltter and enter python path so select the new VE as in image
- After activation, the VR name appears on the left side of the terminal



1.3 How to install in package in VE

- After the VR is activated, we can install packages using pip, and they will be installed exclusively within this environment by writing command (`pip install package name`)
- Pip

Week 2

2.1 Identifiers (Variable Names)

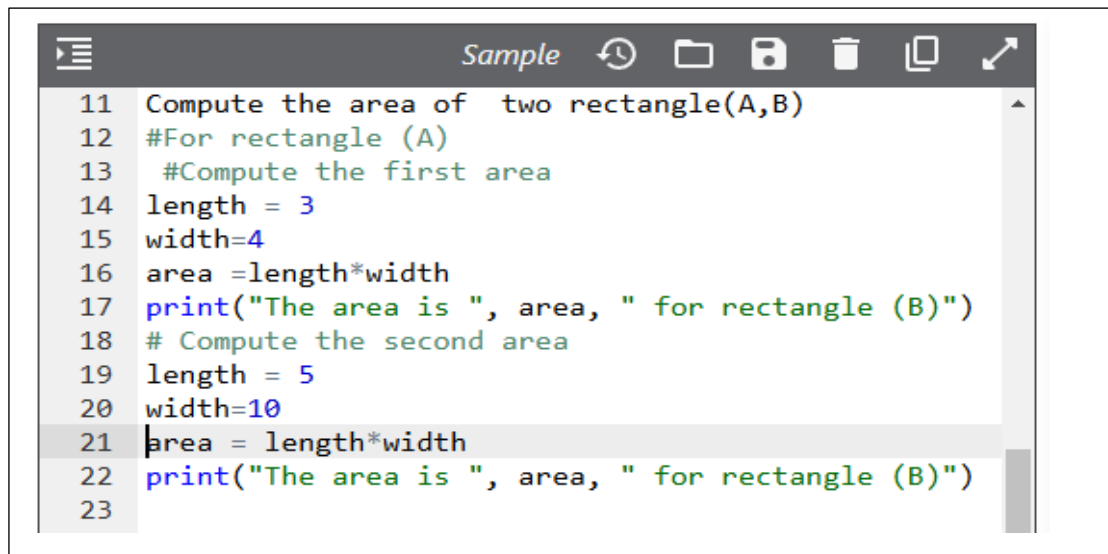
- An identifier is a sequence of characters that consists of letters, digits, underscores (`_`), and asterisk (`*`).
 - An identifier must start with a letter or an underscore. It cannot start with a digit.
 - An identifier cannot contain the e spaces and special characters
 - identifier cannot be a reserved word.
- An identifier can be of any length

✓ my2Var, _temp, car var, data,if,while	reserved word
✓ Names1, total_students ✓ \$Names , total_students	spaces and special characters

✘

2.2 Variable

- A variable is a named place in the memory where a programmer can store data and later retrieve the data using the variable “name”
- Programmers get to choose the names of the variables



```
Sample
11 Compute the area of two rectangle(A,B)
12 #For rectangle (A)
13 #Compute the first area
14 length = 3
15 width=4
16 area =length*width
17 print("The area is ", area, " for rectangle (B)")
18 # Compute the second area
19 length = 5
20 width=10
21 area = length*width
22 print("The area is ", area, " for rectangle (B)")
23
```

2.3 Number Data Type

- Numbers have two main types
- Integers are whole numbers: -14, -2, 0, 1, 100, 401233
- Floating Point Numbers have decimal parts: -2.5 , 0.0, 98.6, 14.0
- There are other number types -they are variations on float and integer
-

```
1 >>> Y= 10
2 >>> type (Y)
3 <class 'int'>
4 >>> temp1= 98.6
5 >>> type(temp1)
6 < class 'float'>
7 >>> type(19)
8 < class 'int'>
9 >>> type(5.4)
10 < class 'float'>
```

2.4 String

- A string is a sequence of characters. It can in enclosed matching to double quotes (“”).
- The string cannot be changed after creation

```
String1 = "A"
String 2 = "4"
String3 = "Good morning"
Print ("this AI course', String3)
```

2.5 Build in Function

- The build in functions are predefined functions that comes with python. The programmer doesn't need a writing function.
- The function can be used to perform various tasks such as string and Metaethical operation
- The function needs to be called with their argument

2.5.1 Build in string Functions

```
"this AI Course".upper() ----->>>("THIS AI COURSE")
"this AI Course".lower()----->>>("this ai course")
Old_text="this week AI" ----->>> ("this is new AI")
Old_text. Replace("new","old")
Old_string=(" 123 ,Almansore st") ----- Old_string (123,Almansore st )
string. strip(Old_string) ----- It is used remove as leading/trailing characters
Old_text.find(AI) ----- 10 - It give the first index occurrence of
character
```

Week 3

3.1 List & Arrays initialization

- A list : The list is set of collections items that can stores the elements from different Data type'
- In Python, a list is a built-in dynamic array that automatically resizes as elements are added or removed. It can store items of different data types.
- The elements of list can be accessed by using index
- List can be expanded by adding element

Examples:

```
L=[] ----- declare list
My_Number=[1,2,3,4,5] ----- Numeric list
Mix_l=["A","s",true,4,-4,9] ----- Mix list
Print(Mix_l[0]) ----- "A" access to first element
Print(Mix_l[-1] -----9 access to first element
Print(Mix_l.append(40,"K"))
Print(Mix_l(2:5) -----["A",true,4)
Print(mix_l.remove(9)) -----["A","s",true,4,-4,40,"K"]
remove element
```

3.2 IF Statement

- The if statement is condition statement, Python has several types of selection statements:

- one-way if statements,
- two-way if-else statements,
- A nested if statements

- A one-way if statement syntax: `if Boolean-expression:
statement(s)
end if`

- A two way if statement syntax: `if Boolean-expression:
statement(s)-for-the-true-case
else:
statement(s)-for-the-false-case
end if`

- Nested if statement syntax: `if Boolean-expression:
statement(s)-for-the-true-case
elseif Boolean—expression
statement(s)-for-the-false-case
else:
statement(s)-for-last if -false-case`

Example1(A one-way if statement):

```
if radius >= 0:
    area = radius * radius * 3.14159
    print ("The area for the circle of radius",
           radius, "is ", area)
Else if
```


Example2(two way if statement):

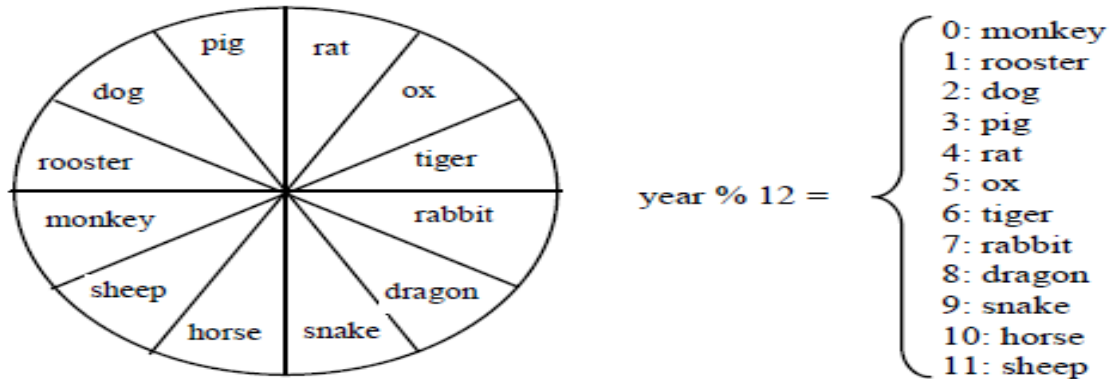
```
If radius >= 0:
    area = radius * radius * math.pi
    print ("The area for the circle of radius", radius, "is",
area)
else:
    print ("Negative input")
End if
```

Example3(Nested if statement):

```
if score >= 90.0:
    grade = 'A'
elif score >= 80.0:
    grade = 'B'
elif score >= 70.0:
    grade = 'C'
elif score >= 60.0:
    grade = 'D'
else:
    grade = 'F',
end if.
```

Home work

- ✓ Q1-Please write python code to find out the Chinese Zodiac sign for a given year. The Chinese Zodiac sign is based on a 12-year cycle, each year being represented by an animal: rat, ox, tiger, rabbit, dragon, snake, horse, sheep, monkey, rooster, dog, and pig, in this cycle.



✓ Q2-Please write python code to classify the interpretation of Body Mass Index (BMI) for people 16 years or older is as follows

BMI	Interpretation
Below 18.5	Underweight
18.5-24.9	Normal
25.0-29.9	Overweight
Above 30.0	Obese

3.3 Loop

- Loop in python is used to execute the code many times under specific conditions. There are two types of loops in python
- ✓ **While Loop:** The while loop is executed since the condition is true
- ✓ **For Loop:** The for loop iterates a specific number of times

While loop syntax:

```
while loop-condition:
    #Loop body
```

Statement

For loop syntax:

for variable in sequence:

#Loop body

Statement

- Break keyword is used to stop the code and continue keyword is used to skip the current step and move to the next step

While Loop Examples

Example1(simple while loop):

```
count = 0
While count < 5:
    print("Programming is fun!")
    count = count + 1
```

Example2 (while loop with user input)

```
inputt = ""
while inputt!= "stop":
    inputt =input ("Enter your input (type 'exit' to stop): ")
    print ("You entered:", inputt)
```

Example3 (while loop with break)

```
sum = 0
number = 0
while number < 20:
    number =number+1
    sum = number *2
    if sum >= 30:
        break
print("The number is ", number)
print("The sum is ", sum)
```

Example 4(while loop with continue)

```
sum = 0
number = 0
while (number < 20):
    number += 1
    if (number == 10 or number == 11):
        continue
    sum = number *2
print("The final sum is ", sum)
```

Homework

Write the python code to solve the following problem

Q1- Suppose that the tuition for a university is \$10,000 this year and tuition increases 7% every year. In how many years will the tuition be doubled?

Q2-Suppose that the restaurants has yearly profits 5000\$ this year and profit increases 5% every year. How much the target profit for four years will be?

Week 4

4.1 Function

- A function is a collection of statements that are grouped together to perform an operation.
- A function contains a header and body. The header begins with the def keyword, followed by function's name and parameters, followed by a colon.
- The variables defined in the function header are known as formal parameters.
- When a function is invoked, you pass a value to the parameter. This value is referred to as actual parameter or argument.
- This program demonstrates calling a function max to return the largest of the int values
- A function that does not return a value is known as a void function in other programming languages such as Python, C++, and C#.

Example : -Write a function to calculate the percentage of successful students in a school with three classes (A, B, and C):

Class A has 50 students, and half of them pass the exam.

Class B has 40 students, and 20% of the students fail the exam.

Class C has 50 students, and 40 students pass the exam. The function should compute and return the overall success rate (percentage of students who passed) in the school

Solution:

```
def calculate_success_rate(class_a_total, class_b_total, class_c_total):  
    # class A  
    class_a_total=50  
    class_a_success =class_a_total*0.5  
    # class B  
    class_b_total=40  
    class_b_success=class_b_total*0.8  
    #class c  
    class_c_total=50  
    class_c_success=40/50  
    total_students =class_a_total +class_b_total + class_c_total  
    total_success =class_a_success+class_b_success + class_c_success  
    per_success_rate =total_success/total_students*100  
    return(per_success_rate)
```

Homework

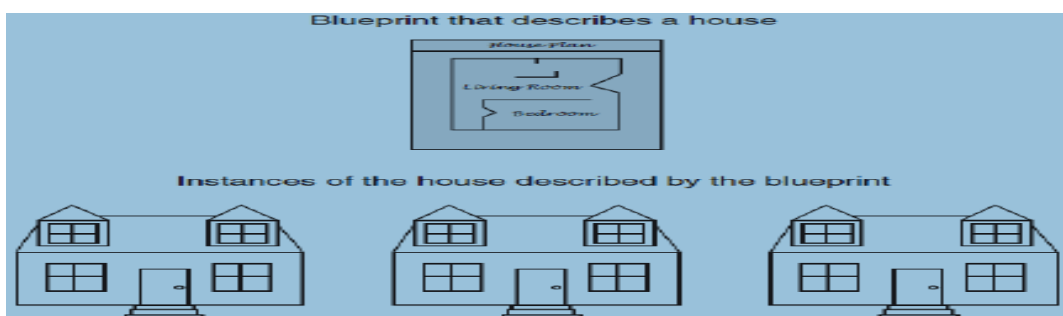
Q1- Write a function to calculate the mean and the stander deviation of a company employers annual salary increment for five years? Assume the total number of employees is 60 to be classified into three categories:

15 employs in group (A) with salary 1500\$,

25 employs in group (B) with salary 1000\$,
20 employs in group (C) with salary 750\$,
Be aware that the annual increment is 0.05.

4.2 Classes

- Class: code that specifies the data attributes and methods of a particular type of object
- Similar to a blueprint of a house or a cookie cutter
- It is simply a template that we construct objects from it.
- Instance: an object created from a class
- Similar to a specific house built according to the blueprint or a specific cookie
- There can be many instances of one class
- self is a parameter that represents an object.
- self-parameter is required in every method in the class –references the specific object that the method is working on.



4.2.1 Classes Syntax with Example

Class Syntax :class ClassName):

```
def __init__(self, param1, param2,.....param(n)):  
    self. param1 = param1
```

```
self. param2 = param2
```

Simple class Examples

```
class books(hard_copy, soft_copy):
    def __init__(self, hard_copy, soft_copy):
        self.hard_copy = hard_copy
        self.soft_copy = soft_copy
book1=books("AI book", "GAI book")
```

Note:

- ✓ (`__init__`) is a special function that is called automatically inside the class when creating an object. It is used for initializing variables.
- ✓ (`self`) refers to the instance of the class self representation of the current instance of the class. It allows variables and methods accessing inside the class.

Examples 2: (class CreditCard)

```
class CreditCard():
    def __init__(self, customer, bank, acct, limit):
        self._customer = customer
        self._bank = bank
        self._account = acct
        self._limit = limit
        self._balance = 0
```

```
"""Return the name of the customer."""
```

```
def get_customer(self):
    return self._customer
```

```
"""Return the bank's name."""
```

```
def get_bank(self):
    return self._bank
```

```
"""Return the card identifying number (typically stored as a string)."""
```

```
def get_account(self):
    return self.get_account
```

```
"""Return current credit limit."""
```

```
def get_limit(self):
```



```

        return self._limit
"""Return current balance."""
    def get_balance(self):
        return self._balance
"""Charge given price to the card, assuming sufficient credit
limit. Return True if charge was processed; False if charge was
denied."""
def charge(self, price)
    if price + self._balance > self._limit:
        return False
    else:
        self._balance += price
        return True
#Excute the code
# enter the inputs
    bank = input("Your banck name: ")
customer= input("Your name: ")
limit = input("Your balance: ")
acct = input("Your account number: ")

#call the class and functions
res=CreditCard(customer,bank,acct,limit)
res1=res.get_customer( )
print (res1)
Res2= res.get_bank()
print(res2)
Res3= res.get_limit()
Res4=res.charge()

```

Homework

Q1-Please write the python class to a create BMI object with the specified name, weight, height, and a default age (40).Class has the inputs with (following name: string, -weight: float,-height: float.-age: int)

A- Returns the BMI

B- Returns the BMI status

Note: The BMI is calculated BY : $BMI = \text{Weight(KG)} / \text{Height(M)}$

There are four statues for adult age 40: For adults, including those aged 40, BMI statues are:

Underweight: Below 18.5

Normal weight: 18.5 – 24.9

Overweight: 25 – 29.9

Obese: 30 and above

4.3 Classes Inheritance

- Parent class is the class being inherited from, also called base class. Child class is the class that inherits from another class, also called derived class.
- A superclass provides common functionalities, while subclasses extend or specialize these functionalities.
- `super()` function that will make the child class inherit all the methods and properties from its parent:

-

Examples 3: (Classes Inheritance)

```
class PredatoryCreditCard(CreditCard):
```

```
    def __init__(self, customer, bank, acct, limit, apr):
        super().__init__(customer, bank, acct, limit)
        self._apr = apr
    def charge(self, price):
        success = super().charge(price)
        if not success:
            self._balance += 5 # charge a $5 fee
        return success
```

```
"""Assess monthly interest on outstanding balance."""
```

```
def process_month(self):  
    if self._balance > 0:  
        monthly_factor = pow(1 + self._apr, 1/12)  
        self._balance *= monthly_factor
```

Examples 4: (Classes Inheritance BMI class)

Please write the child class to a create BMI object with the name(Child_BMI). Compute Lean Body Mass (LBM) for gender
Consider the following

- ✓ man=LBM=(0.407×weight) +(0.267×height(cm))−19.2
- ✓ woman=LBM=(0.252×weight)+(0.473×height(cm))−48.3

parent class:

```
class Bmi_calcaution():
    def __init__(self,name,age,weight,height):
        self.name = name
        self.age = age
        self.weight = weight
        self.height = height
    def get_name(self):
        return self.name
    def get_age(self):
        return self.age
    def get_weight(self):
        return self.weight
    def get_height(self):
        return self.height
```

Enter the input

```
Name= input("Your name : ")
age= int(input("Enter your age (default is 40, press Enter to
skip): ") or 40)
weight =float( input("Enter your weight: "))
height = float(input("Enter your height in CMr: "))
```

call the class and functions

```
res=Bmi_calcaution(name,age,weight,height)
res1=res.get_name( )
res2=res.get_age( )
res3=res.get_weight( )
res4=res.get_height( )
res5=res.bmi_cal( )
print(res5)
res6= res.get_bmi_status( )
print (res6)
```

